```
-- TimeConvert.Mesa  Edited by Sandman on May 12, 1978  3:28 PM

DIRECTORY
  InlineDefs: FROM "inlinedefs" USING [
    DIVMOD, LDIVMOD, LongCARDINAL, LongMult],
  Mopcodes: FROM "mopcodes" USING [zDADD],
  StringDefs: FROM "stringdefs" USING [AppendChar, AppendString],
  TimeDefs: FROM "timedefs" USING [
    BaseYear, currentParameters, currentTime, DaysInFourYears, DefaultTime,
    HardwareTime, PackedTime, StartWeekDay, UnpackedTime, WestEast];

DEFINITIONS FROM TimeDefs;

TimeConvert: PROGRAM IMPORTS StringDefs EXPORTS TimeDefs SHARES TimeDefs =
  BEGIN

  UP: TYPE = POINTER TO UnpackedTime;


  DivideTime: PROCEDURE [num: PackedTime, den: CARDINAL]
    RETURNS [quotient: PackedTime, remainder: CARDINAL] =
    BEGIN OPEN InlineDefs;
    t: CARDINAL;
    [quotient.highbits, t] ← LDIVMOD[num.highbits,0,den];
    [quotient.lowbits, remainder] ← LDIVMOD[num.lowbits,t,den];
    RETURN
    END;

  MultiplyTime: PROCEDURE [multiplicand: PackedTime, multiplier: CARDINAL]
    RETURNS [result: PackedTime] =
    BEGIN OPEN InlineDefs;
    t: CARDINAL;
    result.highbits ← multiplicand.highbits * multiplier;
    [result.lowbits, t] ← LongMult[multiplicand.lowbits, multiplier].product;
    result.highbits ← result.highbits + t;
    RETURN
    END;

  AddTime: PROCEDURE [a: PackedTime, b: INTEGER] RETURNS [PackedTime] =
    BEGIN OPEN InlineDefs;
    DAdd: PROCEDURE [x,y: LongCARDINAL] RETURNS [LongCARDINAL]
      = MACHINE CODE BEGIN Mopcodes.zDADD END;
    highb: INTEGER ← IF b<0 THEN -1 ELSE 0;
    RETURN[DAdd[a,[lowbits:b,highbits:highb]]]
    END;

  CurrentDayTime: PUBLIC PROCEDURE RETURNS [PackedTime] =
    BEGIN
    t: HardwareTime ← currentTime↑;
    RETURN[[highbits: t.high, lowbits: t.low]]
    END;

  TP: TYPE = RECORD [
    beginDST, endDST: CARDINAL,
    zone, zoneminutes: INTEGER];

  TimeParameters: PROCEDURE
    RETURNS [p: TP] =
    BEGIN OPEN p;
    direction: WestEast;
    [beginDST: beginDST, endDST: endDST, zone: zone, zoneminutes: zoneminutes, direction: direction] ←
**currentParameters↑;
    IF direction # west THEN
      BEGIN zone ← -zone; zoneminutes ← -zoneminutes END;
    RETURN
    END;

  MonthTable: ARRAY [0..12] OF CARDINAL =
    [0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366];

  UnpackDT: PUBLIC PROCEDURE [p: PackedTime] RETURNS [unp: UnpackedTime] =
    BEGIN
    u: UP = @unp;
    day4, day, yr4: CARDINAL;
    month: CARDINAL ← 1;
    t: InlineDefs.LongCARDINAL;
```

```
    parms: TP;

    parms ← TimeParameters[];
    IF p = DefaultTime THEN p ← CurrentDayTime[];
    [p, u.second] ← DivideTime[p,60];
    p ← AddTime[p,-parms.zoneminutes]; -- ignore underflow
    [p, u.minute] ← DivideTime[p,60];
    u.zone ← parms.zone;
    u.dst ← FALSE;
    p ← AddTime[p,-parms.zone]; -- ignore underflow
    DO -- have to repeat if DST
      [t, u.hour] ← DivideTime[p,24];
      u.weekday ← (t.lowbits + StartWeekDay) MOD 7;
      [yr4,day4] ← InlineDefs.DIVMOD[t.lowbits,DaysInFourYears];
      day4 ← (IF day4 >= 2*365+31+28 THEN 3 ELSE (day4+(365-31-28))/365) + day4;
      [day4, day] ← InlineDefs.DIVMOD[day4, 366];
      u.year ← BaseYear + yr4*4 + day4;
      WHILE day >= MonthTable[month] DO month ← month + 1 ENDLOOP;
      u.month ← month ← month-1;
      day ← day + 1;
      u.day ← day - MonthTable[month];
      IF u.dst OR ~CheckDateGE[u, day, parms.beginDST, 2]
        OR CheckDateGE[u, day, parms.endDST, 1] THEN EXIT;
      p ← AddTime[p,1];
      u.dst ← TRUE;
      ENDLOOP;
    RETURN
    END;

  InvalidTime: PUBLIC ERROR = CODE;

  PackDT: PUBLIC PROCEDURE [unp: UnpackedTime, computeDST: BOOLEAN] RETURNS [PackedTime] =
    BEGIN
    t: PackedTime;
    u: UP = @unp;
    year, month, day, day1, hour, minute, second: CARDINAL;
    zone: INTEGER;
    dst: BOOLEAN;
    yr3: [0..3];
    tp: TP ← TimeParameters[];

    [year: year, month: month, day: day, hour: hour, minute: minute, second: second, zone: zone, dst: d
**st] ← u↑;
    IF (year ← year-BaseYear) >= 136 OR
      month >=12 OR day ~IN[1..31] OR
      hour >= 24 OR minute >= 60 OR second >= 60 THEN ERROR InvalidTime;
    yr3 ← year MOD 4;
    IF day > LOOPHOLE[MonthTable[month+1] - MonthTable[month], CARDINAL] OR
      (month = 1 AND day = 29 AND yr3 # 3) THEN ERROR InvalidTime;

    -- compute days this year in day1
    day1 ← MonthTable[month] + day;
    IF yr3 # 3 AND month >= 2 THEN day1 ← day1-1;

    t.highbits ← 0;
    t.lowbits ← (year/4)*DaysInFourYears + yr3*365 + day1 - 1;
    u.weekday ← t.lowbits MOD 7;
    IF computeDST THEN
      BEGIN OPEN tp;
      IF CheckDateGE[u,day1,beginDST,2]
        AND ~CheckDateGE[u,day1,endDST,2] THEN zone ← zone - 1
      END
    ELSE
      BEGIN
      tp.zone ← zone; tp.zoneminutes ← 0;
      IF dst THEN tp.zone ← tp.zone - 1;
      END;
    RETURN[
      AddTime[MultiplyTime[
      AddTime[MultiplyTime[
      AddTime[AddTime[MultiplyTime[
      t,24],hour],tp.zone],60],minute+tp.zoneminutes],60],second]]
    END;

  AppendDayTime: PUBLIC PROCEDURE [s: STRING, unp: UnpackedTime] =
    BEGIN
```

```
    u: UP = @unp;
    p: CARDINAL ← s.length;
    m: CARDINAL;
    w2d: PROCEDURE [v: CARDINAL] =
      BEGIN
      d1, d2: CARDINAL;
      [d1, d2] ← InlineDefs.DIVMOD[v,10];
      IF d1 # 0 THEN s[p] ← '0 + d1;
      s[p+1] ← '0 + d2;
      p ← p + 3;
      END;

    StringDefs.AppendString[s, " 0-xxx-00  0:00:00"L];
    w2d[u.day];
    m ← u.month*3;
    THROUGH [0..2] DO
      s[p] ← ("JanFebMarAprMayJunJulAugSepOctNovDec"L)[m];
      p ← p + 1; m ← m + 1;
      ENDLOOP;
    p ← p + 1;
    w2d[u.year MOD 100];
    w2d[u.hour];
    w2d[u.minute];
    w2d[u.second];
    RETURN
    END;

  AppendFullDayTime: PUBLIC PROCEDURE [s: STRING, unp: UnpackedTime] =
    BEGIN
    z: INTEGER;
    KnownZones: TYPE = [4..10];
    zones: PACKED ARRAY KnownZones OF CHARACTER ← ['A,'E,'C,'M,'P,'Y,'H];
    AppendDayTime[s,unp];
    IF (z←unp.zone) IN KnownZones THEN
      BEGIN OPEN StringDefs;
      c: CHARACTER ← IF unp.dst THEN 'D ELSE 'S;
      AppendChar[s,' ];
      AppendChar[s,zones[z]];
      AppendChar[s,c];
      AppendChar[s,'T];
      END;
    END;

  CheckDateGE: PROCEDURE [u: UP, days, dstDay, dstHour: INTEGER] RETURNS [BOOLEAN] =
    BEGIN
    weekday: INTEGER ← u.weekday;
    RETURN[
      IF days < dstDay-6 THEN FALSE
      ELSE IF days > dstDay THEN TRUE
      ELSE IF weekday = 6 THEN u.hour >= dstHour
      ELSE days-weekday > dstDay-6]
    END;

  END..
```